Компьютерные технологии в научных исследованиях и образовании

Юрий Анатольевич Богоявленский, заведующий кафедрой Информатики и математического обеспечения, к.т.н., доцент, ybgv@cs.petrsu.ru,

Лекция 2. Пакет GNU Octave. Введение

Общие сведения

Запуск пакета `octave` на выполнение

Примеры работы из командной строки

Исполняемые программы в `octave`

Типы данных

Числовые Типы Данных

Числовые константы

Матрицы

Общие сведения

В разделе «about» документа:

GNU Octave [Электронный ресурс] — URL: https://octave.org/ (15.08.2022)

представлено следующее описание. Цитата.

«GNU Octave - это язык высокого уровня, предназначенный в первую очередь для решения задач численными методами. Он предоставляет удобный интерфейс командной строки для численного решения линейных и нелинейных задач, а также для выполнения других вычислительных экспериментов с использованием языка, который в основном совместим с пакетом Matlab. Он также может быть использован как язык пакетной обработки.

Octave обладает широким набором инструментов для численного решения задач линейной алгебры, нахождения корней нелинейных уравнений, интегрирования обычных функций, манипулирования многочленами и интегрирования обыкновенных дифференциальных и дифференциально-алгебраических уравнений. Он легко расширяется и настраивается с помощью пользовательских функций, написанных на собственном языке Octave, или с помощью динамически загружаемых модулей, написанных на C++, C, Fortran или других языках.

GNU Octave это свободно распространяемое программным обеспечением. Вы можете распространять и/или изменять его в соответствии с условиями GNU General Public License (GPL), опубликованной Фондом свободного программного обеспечения (Free Software Foundation). Октава была разработана Джоном У. Итоном (John W. Eaton) вместе с большим количеством соавторов» (конец цитаты). Octave также имеет хорошо развитые инструменты визуализации.

В исследовательском сообществе Octave активно используется для анализа данных, обработки изображений, компьютерного зрения, экономических исследованиях, интеллектуального анализа данных, статистический анализ, машиннго обучения, обработка сигналов и во многие другие областях.

Пакет активно развивается с 1993 по настоящее (2022 г.) время. Последний выпуск (версия 7.2.0), посвященный исправлению ошибок, был сделан 28 июля 2022 г. Анализ кода, проекта, сделанный в 2020 г. и представленный на ресурсе: https://www.openhub.net/p/octave (15.08.2022) дал следующие данные:

- Выполнено всего 48,443 фиксаций программного кода в репозитории (commits), реализованных 525 разработчиками; объем кода - 1 180 830 строк.
- Большая часть кода написана на языке C++ и содержит разумное число строк комментариев.
- Проект имеет хорошо зарекомендовавшую себя, зрелую кодовую базу, поддерживаемую очень большой командой разработчиков

- По модели оценки трудоемкости COCOMO на разработку было потрачено 333 человеко года.

На ресурсе:

Octave Packages [Электронный ресурс] — URL:

https://gnu-octave.github.io/packages/ (15.08.2022) представлены более девяносто пакетов, расширяющих функциональность Octave. В этом списке например такие пакеты как:

- `symbolic` для символьных преобразований;
- `optim` для задач нелинейной оптимизации:
- `matgeom` для задач 2-х и 3-х мерной геометрии:
- `parallel` для организации параллельных вычислений;
- уже упомянутая нами библиотека GNU Scientific Library (`gsl`).

По ссылке на имя пакета можно получить описания входящих в него функций. На ресурсе:

Octave Forge. Packages. Community packages. [Электронный ресурс] — URL: https://octave.sourceforge.io/packages.php (15.08.2022).

представлен еще один список таких пакетов. Описания функций, входящих в эти пакеты можно получить на ресурсе:

Octave Forge. Function list. [Электронный ресурс] — URL:

https://octave.sourceforge.io/list_functions.php?sort=package

В нашем курсе мы не рассматриваем вопросы взаимоотношения этих списков и не будем изучать входящие в них пакеты.

Документация Octave представлена на ресурсах:

- John W. Eaton, David Bateman, Søren Hauberg, Rik Wehbring (2022). GNU Octave version 7.1.0 manual: a high-level interactive language for numerical computations. [Электронный ресурс]
  — URL https://www.gnu.org/software/octave/doc/v7.1.0/ (16.08.2022)
- [Док] Файл формата pdf [Электронный ресурс] — URL:
  https://docs.octave.org/octave-7.1.0.pdf (16.08.2022)
  Далее ссылки на этот документ будут даны в виде [Док, с. …].
- Questions tagged [octave]. [Электронный ресурс]— URL:

https://stackoverflow.com/questions/tagged/octave (16.08.2022)
- GNU Octave Wiki. [Электронный ресурс] — URL: https://wiki.octave.org/GNU_Octave_Wiki (16.08.2022)

Запуск пакета `octave` на выполнение

Octave кроссплатформенный пакет, процедуры его установки для различных ОС описаны на ресурсе https://octave.org/. Работать с пакетом можно либо из командной строки интерпретатора языка либо через IDE с графическим интерфейсом. Как правило команда `octave` запускает работу в режиме командной строки, команда `octave --gui` — в режиме IDE. Команда `octave <имя_файла>` приведет к выполнению файла (скрипта), содержащего конструкции языка `octave`, выводу результатов их выполнения и завершению работы. Другие ключи управления запуском описаны в [Док, с. 15] и в выводе команды `octave –h`.

Примеры работы из командной строки

Пример цифровых вычислений

```
ybgv@ybgv-home:~> octave
GNU Octave, version 7.1.0
Copyright (C) 1993-2022 The Octave Project Developers.
This is free software; see the source code for copying conditions.

There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-suse-linux-gnu".

Additional information about Octave is available at
https://www.octave.org
.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-
involved.html
```

```
Read https://www.octave.org/bugs.html to learn how to submit bug
reports.
For information about changes from previous versions, type 'news'.

octave:1> # длинный формат
octave:1>
octave:1> format long
octave:2>
octave:2> # вычислить и вывести для комплексного аргумента  e^(pi*i)
octave:2>
octave:2> exp (i*pi)
ans = -1.000000000000000e+00 + 1.224646799147353e-16i
octave:3>
octave:3> # короткий формат
octave:3>
octave:3> format short
octave:4>
octave:4> exp (i*pi)
ans = -1.0000e+00 + 1.2246e-16i
octave:5>
octave:5> # округлить до ближайшего целого
octave:5>
octave:5> round(exp (i*pi))
ans = -1
octave:6>
octave:6> # решение матричного уравнения A·X=B
octave:6>
octave:6> # создать матрицу и не выводить (есть ; в конце  оператора)
octave:6>
octave:6> A=[ 2 3;-2 6];
octave:7>
octave:7> # и вывести матрицу
octave:7>
octave:7> A
A =

  2   3
 -2   6

octave:8>
octave:8> # создать матрицу и вывести (нет ; в конце оператора)
octave:8>
octave:8> B=[2 5;2/3 5/3]
B =

  2.0000   5.0000
  0.6667   1.6667
```

```
octave:9>
octave:9> # решить первым способом
octave:9>
octave:9> X=A\B
X =

  0.5556   1.3889
  0.2963   0.7407

octave:10>
octave:10> # решить вторым способом
octave:10>
octave:10> X=inv(A)*B
X =

  0.5556   1.3889
  0.2963   0.7407

octave:11>
octave:11> # проверить решение
octave:11>
octave:11> A*X-B
ans =

           0   8.8818e-16
 -1.1102e-16  -6.6613e-16

octave:12>
```

Пример символьных вычислений

```
   octave:2> # загрузить пакет symbolic
   octave:2>
   octave:2> pkg load symbolic
   octave:3>
   octave:3> # вывести список пакетов
   octave:3>
octave:3> pkg listName  | Version | Installation directory
--------------+---------+-----------------------
         gsl  |   2.1.1 | /usr/share/octave/packages/gsl-2.1.1
```

```
      symbolic *|    3.0.0 | /usr/share/octave/packages/symbolic-
3.0.0
octave:4>
octave:4> # определить переменные символьного типа
octave:4>
octave:4> syms x y z
Symbolic pkg v3.0.0: Python communication link active, SymPy
v1.5.1.
octave:5>
octave:5> # определить символьный полином
octave:5>
octave:5> p = x^3 + x^2 + x + 1
p = (sym)

   3    2
  x  + x  + x + 1

octave:6>
octave:6> # получить для него разложение по схеме Горнера
octave:6>
octave:6> q = horner (p)
q = (sym) x·(x·(x + 1) + 1) + 1
octave:7>
octave:7> # определить символьное выражение
octave:7>
octave:7> y=(sqrt(x)+1)*(sqrt(x)-1)+(x-1)*(x-1)*(x-1)
y = (sym)

                              3
  (√x - 1)·(√x + 1) + (x - 1)

octave:8>
octave:8> # упростить его
octave:8>
octave:8> simplify(y)
ans = (sym)

   3      2
  x  - 3·x  + 4·x - 2

octave:9>
octave:9> # определить символьную функцию от x
octave:9>
octave:9> f = sin (cos (x))
f = (sym) sin(cos(x))
octave:10>
octave:10> # найти формулу ее производной по x
```

```
octave:10>
octave:10> diff (f, x)
ans = (sym) -sin(x)·cos(cos(x))
octave:11>
octave:11> # определить другую символьную функцию от x
octave:11>
octave:11> f=(5*sin(2*x))/sqrt(cos(2*x))
f = (sym)

   5·sin(2·x)
  ─────────────

    ──────────
  ╲╱ cos(2·x)

octave:12>
octave:12> # найти формулу ее производной по x
octave:12>
octave:12> diff (f, x)
ans = (sym)

        2
  5·sin (2·x)          ──────────
  ──────────── + 10·╲╱ cos(2·x)
     3/2
  cos   (2·x)

octave:13>
octave:13> # присвоить символьной переменное значение этой формулы
octave:13>
octave:13> df = diff (f, x)
df = (sym)

        2
  5·sin (2·x)          ──────────
  ──────────── + 10·╲╱ cos(2·x)
     3/2
  cos   (2·x)

octave:14>
```

Команды `exit` или `quit` завершают работу `octave`. В [Док, с. 19] описаны вариации этх команд.

В [Док, с. 25] даны инструкции по редактированию командной строки. В большинстве случаев достаточно стрелок на клавиатуре (горизонтальные

передвигают курсор по символам команды, вертикальные по выполненным командам) и клавиш «Backspace» и «Delete».

Справочные сведения по элементам octave можно получить с помощью команды help, а описание ее вариаций по команде help help. Например

команд help --list выведет список всех операторов, ключевых слов, встроенных и загружаемых функций, доступных в текущем сеансе octave. Команда help <имя> выведет сведения об элементе, имеющем это имя. Команда doc выведет графическое окно, обеспечивающее поиск и просмотр документации octave. В [Док, с. 21] подробно описаны способы получения справочной информации.

Диалог текущей сессии octave будет записан в файл с именем diary в текущем каталоге по команде diary on. Конструкции octave в выполняемых файлах будут выводиться после их запуска по команде echo on, отмена вывода произойдет по команде echo off. Описание команд diary и echo дано в [Док, с. 36].

В [Док, с. 36] описаны два вида выводимых системой сообщений об ошибках.

В окнах интегрированной среды разработки (IDE) octave с графическим интерфейсом реализованы:

- интерпретатор языка octave, работающий в режиме командной строки;
- область переменных, где перечислены созданные в сеансе работы переменные и их типы;
- журнал выполненных команд;
- редактор кода с подсветкой синтаксиса и встроенным отладчиком;
- документация с возможностью поиска;
- редактор переменных;
- встроенный диспетчер файлов.

IDE целесообразно использовать при отладке программ. Позже мы рассмотрим работу в IDE.

Исполняемые программы в octave и механизм «шебанг»

Существует два вида исполняемых программ — функции и скрипты. Файлы, содержащие эти программы для совместимости с matlab имеют расширение .m. Файлы функций позволяют определить функции пользователя, которые используются так же как встроенные функции octave. Работа с функциями будет рассмотрена позже.

Скриптом, вообще говоря, называют файл, содержащий конструкции любого языка, имеющего интерпретатор. В нашем случае строки файла скрипта octave будут выполняться точно так же как при их вводе в режиме командной строки. Запуск скрипта на выполнение обычно происходит по команде вида:

<имя_интерпретатора> <имя_скрипта>.

Как правило в GNU и Unix системах реализован универсальный механизм запуска скриптов по команде вида:

<имя_скрипта>

без указания в ней имени исполняемого файла интерпретатора, которое, в этом случае, задается в первой строке файла скрипта. Этот механизм называется «шебанг» (англ. shebang, sha-bang, hashbang, pound-bang, hash-pling). Он реализован путем размещения в первой строке файла скрипта последовательность из двух символов — решётки и восклицательного знака "#!" за которой следует имя исполняемого файла интерпретатора, на языке которого написан скрипт.

При этом после присвоения файлу скрипта права доступа «запуск» он будет выполняться путем указания в командной строке только его имени без указания имени интерпретатора. В нашем случае файл скрипта должен содержать конструкции языка octave, а его первая строка должна иметь вид:

```
#! /usr/bin/octave -qf
```

если исполняемый файл octave находится в каталоге /usr/bin. Если имя файла скрипта — script1, находящегося в текущем каталоге, то в ОС Unix интерпретатор octave начнет его выполнение по команде ./script1.

Комментарием в строке скрипта считается текст от символа «#» или «!» до конца строки. Комментарий из нескольких строк это текст между строками «#{» и «#}» или «!{» и «!}».

Типы данных

В octave определен богатый набор встроенных типов данных, соответствующих основным математическим объектам. Это вещественные и комплексные скаляры и матрицы, диапазоны, символьные строки, структуры данных и массивы ячеек (cell array), элементы которых могут иметь вышеперечисленные типы. В octave определены следующие классы данных, которые соответствуют архитектурным типам данных (кроме логического и комплексных классов):

"logical"	"int64"	"double"
"char"	"uint8"	"single"
"int8"	"uint16"	"double complex"
"int16"	"uint32"	"single complex"
"int32"	"uint64"	

Классы intNN соответствуют NN битовым представлениям знаковых целых, uintNN - NN битовым представлениям беззнаковых целых.

Выражения в octave определены аналогично другим языкам, позже мы их рассмотрим. Будем далее произвольное выражение обозначать символами «expr». Класс выражения можно определить вызвав функцию class(expr).

Список встроенных типов данных, определенных как на основе этих классов, так и из соображений объектно-ориентированной реализации, выводится по команде `typeinfo()` и для версии `7.1.0` имеет вид:

<unknown type>	uint32 scalar	dynamically-linked function
cell	uint64 scalar	function handle
scalar	int8 matrix	float scalar
complex scalar	int16 matrix	float complex scalar
matrix	int32 matrix	float matrix
diagonal matrix	int64 matrix	float diagonal matrix
complex matrix	uint8 matrix	float complex matrix
complex diagonal matrix	uint16 matrix	float complex diagonal matrix
range	uint32 matrix	permutation matrix
double_range	uint64 matrix	magic_int
bool	sparse bool matrix	magic_uint
bool matrix	sparse matrix	null_matrix
string	sparse complex matrix	null_string
sq_string	struct	null_sq_string
int8 scalar	scalar struct	lazy_index
int16 scalar	class	onCleanup
int32 scalar	cs-list	octave_java
int64 scalar	magic-colon	object
uint8 scalar	built-in function	
uint16 scalar	user-defined function	

Тип выражения выводится по команде `typeinfo(expr)`. В таблице 58 типов (версия 7.1.0) основные из которых описывают:

- скаляры, соответствующие архитектурным типам данных и комплексным числам;
- матрицы, тип элементов которых есть один их типов скаляров;
- матрицы с различными матричными свойствами;
- булевские и строковые данные;

Определены также типы,описывающие функции и другие типы.

В [Док, с. 41-44] описаны функции работы с типами, в том числе их преобразования.

Числовые Типы Данных

Далее 64 битовые числа с плавающей запятой двойной точности будем обознчать аббревиатурой Fl64.

По умолчанию значения данных этих типов определяются, в большинстве случаев, как Fl64, исключения из этого правила будут описываться явно. Для платформ, использующих стандарт IEEE 754 с плавающей запятой, значения Fl64 находятся в диапазоне приблизительно от $2.2251e^{-308}$ до $1.7977e^{+308}$, а относительная точность составляет приблизительно 2.2204^{e-16}, т.е. можно ожидать, что в лучшем случае 15 десятичных цифр будут правильными. Точные значения заданы встроенными переменными `realmin, realmax` и `eps` соответственно.

Определить данные других типов нужно путем явного указания.

Пример.

```
octave:27> format long
octave:28> realmin
ans = 2.225073858507201e-308
octave:29> realmax
ans = 1.797693134862316e+308
octave:30> eps
ans = 2.220446049250313e-16
octave:31> x=5
x = 5
octave:32> typeinfo(x)
ans = scalar
```

```
octave:33> class(x)
ans = double
octave:34> y=[2.3 3.4; 5.6 6.7]
y =

   2.300000000000000   3.400000000000000
   5.600000000000000   6.700000000000000

octave:35> typeinfo(y)
ans = matrix
octave:36> class(y)
ans = double
octave:37> m=int32(16000)
m = 16000
octave:38> typeinfo(m)
ans = int32 scalar
octave:39> class(m)
ans = int32
octave:40> n=int32([2 3;4 5])
n =

  2  3
  4  5

octave:41> typeinfo(n)
ans = int32 matrix
octave:42> class(n)
ans = int32
octave:43> whos
Variables visible from the current scope:

variables in scope: top scope

  Attr   Name        Size                     Bytes  Class
  ====   ====        ====                     =====  =====
         ans         1x5                          5  char
         m           1x1                          4  int32
         n           2x2                         16  int32
         x           1x1                          8  double
         y           2x2                         32  double

Total is 15 elements using 65 bytes

octave:44>
```

Числовые константы

Первые два символа константы указывабт на основание системы счисления в которой она задана:

десятичные цифры — 10;

`0x` или `0X` — 16;

`0b` или `0B` — 2.

Для наглядности цифры можно разделять символом подчеркивания '_', который игнорируется интерпретатором `octave`.

Примеры задания константы 42.

```
42 # десятичная система счисления
0x2A # шестнадцатеричная система счисления
0b101010 # двоичная система счисления
0b10_1010 # двоичная система счисления
```

Только для десятичной системы счисления можно задать константу задать как десятичную дробь или в экспоненциальной (научной форме).

Примеры задания константы 0.105:

```
.105
1.05e-1
.00105e+2
```

Числовые константы по умолчанию имеют значения Fl64.

Для определения констант других архитектурных типов нужно применять соответствующие функции.

Пример.

```
octave:1> x=int32(100)
x = 100
octave:2> y=int32(0x64)
y = 100
octave:3> class(x)
```

```
ans = int32
octave:4> class(y)
ans = int32
octave:5>
```

Комплексные константы представляются в виде условной суммы вещественной части и мнимой частей, которая обозначается действительным числовым значением со следующим за ним без пробела индикатором комплексной части (‘i’, ‘j’, ‘I’ или ‘J’, который представляет мнимую единицу √-1.

Пример эквивалентных определений одной и той же комплексной константы:

```
3 + 42i, 3 + 42j, 3 + 42I, 3 + 42J, 3.0 + 42.0i
3.0 + 0x2Ai, 3.0 + 0b10_1010i, 0.3e1 + 420e-1i
```

Для определения комплексной константы можно использовать функцию `complex`.

Пример.

```
octave:16> complex(3)
ans =  3 + 0i
:17> complex(3,10)
ans =  3 + 10i
```

Вещественная и мнимая части комплексной константы по умолчанию имеют значения Fl64.

Матрицы

В `octave` определены интуитивные средства задания матриц. Символы квадратных скобок `[` и `]` открывают и закрывают определение матрицы,

функция size (expr) выводит количество строк и столбцов. По умолчанию элементы матриц имеют тип Fl64.

Примеры:

```
ybgv@ybgv-home:~> octave -q
octave:1> A=[11,12,13;21,22,23;31,32,33]   # задать матрицу
A =

   11   12   13
   21   22   23
   31   32   33

octave:2> size(A),typeinfo(A)   # ее размер и тип
ans =

   3   3

ans = matrix
octave:3> size(A),typeinfo(A),class(A)   # ее размер, тип, класс
ans =

   3   3

ans = matrix
ans = double
octave:4> B=[A,A]   # строка матрици состоит и матриц
B =

   11   12   13   11   12   13
   21   22   23   21   22   23
   31   32   33   31   32   33

octave:5> size(B),typeinfo(B),class(B)   # ее размер, тип, класс
ans =

   3   6

ans = matrix
ans = double
octave:6> C=[A;A]   # столбцы матрицы есть матрицы
C =

   11   12   13
   21   22   23
   31   32   33
   11   12   13
```

```
   21   22   23
   31   32   33

octave:7> size(C),typeinfo(C),class(C)   # ее размер, тип, класс
ans =

   6   3

ans = matrix
ans = double
octave:8>  D=int32([11,12,13;21,22,23;31,32,33])       #   задать
целочисленную матрицу
D =

  11  12  13
  21  22  23
  31  32  33

octave:9> size(D),typeinfo(D),class(D)    # ее размер, тип, класс
ans =

   3   3

ans = int32 matrix
ans = int32
octave:10> E=[1+2i,3,4;5,6,6]    # задать комплексную матрицу
E =

   1 + 2i   3 + 0i   4 + 0i
   5 + 0i   6 + 0i   6 + 0i

octave:11> size(E),typeinfo(E),class(E)  # ее размер, тип, класс
ans =

   2   3

ans = complex matrix
ans = double
octave:12> format short e   # экспоненциальный формат вывода
octave:13> B
B =

 Columns 1 through 5:

   1.1000e+01   1.2000e+01   1.3000e+01   1.1000e+01   1.2000e+01
   2.1000e+01   2.2000e+01   2.3000e+01   2.1000e+01   2.2000e+01
   3.1000e+01   3.2000e+01   3.3000e+01   3.1000e+01   3.2000e+01
```

```
 Column 6:

   1.3000e+01
   2.3000e+01
   3.3000e+01

octave:14> C
C =

   1.1000e+01   1.2000e+01   1.3000e+01
   2.1000e+01   2.2000e+01   2.3000e+01
   3.1000e+01   3.2000e+01   3.3000e+01
   1.1000e+01   1.2000e+01   1.3000e+01
   2.1000e+01   2.2000e+01   2.3000e+01
   3.1000e+01   3.2000e+01   3.3000e+01

octave:15> E
E =

 Columns 1 and 2:

   1.0000e+00 + 2.0000e+00i   3.0000e+00 +          0i
   5.0000e+00 +          0i   6.0000e+00 +          0i

 Column 3:

   4.0000e+00 +          0i
   6.0000e+00 +          0i

octave:16> whos    # характеристики всех переменных
Variables visible from the current scope:

variables in scope: top scope

  Attr   Name        Size                     Bytes  Class
  ====   ====        ====                     =====  =====
         A           3x3                         72  double
         B           3x6                        144  double
         C           6x3                        144  double
         D           3x3                         36  int32
  c      E           2x3                         96  double
         ans         1x6                          6  char

Total is 66 elements using 498 bytes

octave:17> format free
```

```
octave:19> B
B =

 11 12 13 11 12 13
 21 22 23 21 22 23
 31 32 33 31 32 33

octave:20> D
D =

  11  12  13
  21  22  23
  31  32  33


octave:21> E
E =

 (1,2) (3,0) (4,0)
 (5,0) (6,0) (6,0)
```